Getting started with EPICS on RTEMS

W. Eric Norum

March 6, 2003

Contents

Chapter 1

Introduction

This tutorial presents the steps needed to obtain and install the development tools and libraries required to run EPICS IOC applications using RTEMS. As you can see by the size of this document the process isn't trivial, but it's not terribly difficult either.

Chapter two deals with the problem of getting all the tools in place. This is the most difficult task. Once the tools and operating system are working most of the work is complete. Chapter three shows the steps needed to configure and build your first EPICS application for RTEMS. After you've completed those steps you can forget about this document and use the generic EPICS documentation.

This is a living document. Please let me (norume@aps.anl.gov) know which of these instructions worked for you and which did not.

Chapter 2

Infrastructure – Tools and Operating System

2.1 Create the RTEMS source and installation directories

There should be at least 300 Mbytes of space available on the drive where these directories are located. The suggested location for the installation directory is `/opt/rtems`. If you must place the root of the RTEMS installation in some other location, it is still a good idea to make a symbolic link from `/opt/rtems` to the actual root of the RTEMS installation tree. The location of the RTEMS source is not critical. This document assumes that the root of the RTEMs source tree is `/usr/local/source/RTEMS`.

Create the directories where the source will be placed and the results of the build installed:

```
/usr/local/source/RTEMS
/usr/local/source/RTEMS/tools
/opt/rtems
```

2.2 Add the directory containing the tools to your shell search path

The following sections assume that the directory into which you will install the cross-development tools (`/opt/rtems/bin`) is on your shell search path. For shells like sh, bash, zsh and ksh you can to this with

```
PATH="$PATH:/opt/rtems/bin"
```

For shells like csh and tcsh you can

```
set path = ( $path /opt/rtems/bin )
```

2.3 Get and build the development tools

RTEMS uses the GNU toolchain to build the executive and libraries. Information about the GNU tools can be found on the GNU home page. If you're feeling brave you can skip the following sections and turn loose the script included in appendixA. The script attempts to download, unpack, configure, build and install the GNU cross-development tools and libraries for one or more target architectures. To use the script, edit the `ARCH="i386"` line to reflect the architectures you wish to support, then

```
sh getAndBuildTools.sh
```

Set the MAKE environment variable to the name of whatever make program you need for your system.

2.3.1 Download the tool source files

The source for the GNU tools should be obtained from the On-line Applications Research (OAR) FTP server since that server provides any RTEMS-specific patches that may have to be applied before the tools can be built.

The files in the OAR FTP server directory ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source should be downloaded to the RTEMS/tools directory created above. The files can be downloaded using a web browser or a command-line program such as `curl` or `wget`. (note that the command examples have been split to help them fit on the page):

```
curl --remote-name
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/binutils-2.13.2.1.tar.bz2
curl --remote-name
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/gcc-3.2.1.tar.gz
curl --remote-name
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/newlib-1.11.0.tar.gz
curl --remote-name
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/gcc-3.2.1-rtems-20021209.diff
curl --remote-name
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/newlib-1.11.0-rtems-20020203.diff
```

or

```
wget --passive-ftp --no-directories --retr-symlinks
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/binutils-2.13.2.1.tar.bz2
wget --passive-ftp --no-directories --retr-symlinks
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/gcc-3.2.1.tar.gz
wget --passive-ftp --no-directories --retr-symlinks
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/newlib-1.11.0.tar.gz
wget --passive-ftp --no-directories --retr-symlinks
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/gcc-3.2.1-rtems-20021209.diff
wget --passive-ftp --no-directories --retr-symlinks
    ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source/newlib-1.11.0-rtems-20020203.diff
```

Depending on the type of firewall between your machine and the OAR FTP server you may need to remove the `--passive-ftp` option from the `wget` commands.

2.3.2 Unpack the source archives:

The following commands will extract the GNU tool sources from the downloaded tar archive files.

```
bzcat binutils-2.13.2.1.tar.bz2 | tar xf -
zcat gcc-3.2.1.tar.gz | tar xf -
zcat newlib-1.11.0.tar.gz | tar xf -
```

To build the newlib libraries needed by RTEMS you must make a symbolic link to the newlib source directory from the gcc source directory.

```
cd gcc-3.2.1
rm -rf newlib
ln -s ../newlib-1.11.0/newlib newlib
cd ..
```

2.3.3 Apply any RTEMS-specific patches

If any patch files (those with a `.diff` suffix) were downloaded from the OAR FTP server the patches in those files must be applied before the tools can be compiled. For example, here is how the patches can be applied to the gcc sources:

```
cd gcc-3.2.1
patch -p1 <../gcc-3.2.1-rtems-20021209.diff
cd ..
```

And here is how the patches can be applied to the newlib sources:

```
cd newlib-1.11.0
patch -p1 <../newlib-1.11.0-rtems-20020203.diff
cd ..
```

2.3.4 Configure, build and install the ’binutils’:

The commands in this section must be repeated for each desired target architecture. The examples shown build the tools for Intel ’x86 targets.

1. Create a directory in which the tools will be built and change to that directory.

   ```
   rm -rf build
   mkdir build
   cd build
   ```

2. Configure the tools.

   ```
   ../binutils-2.13.2.1/configure --target=i386-rtems --prefix=/opt/rtems
   ```

   You should replace the ’`i386`’ with the name of the architecture for which you’re building the tools. Common alternatives are ’`m68k`’ and ’`ppc`’ for the Motorola M68k and Power-PC family of processors, respectively.

3. Build and install the tools.

   ```
   make -w all install
   ```

   In this and all subsequent cases the use of a GNU make program is required. On some hosts you’ll have to use `gmake` instead of `make`.

4. Return to the directory containing the tool and library sources.

   ```
   cd ..
   ```

2.3.5 Configure, build and install the cross-compiler and libraries

1. Create a directory in which the tools will be built and change to that directory.

   ```
   rm -rf build
   mkdir build
   cd build
   ```

2. Configure the compiler and libraries.

   ```
   ../gcc-3.2.1/configure --target=i386-rtems --prefix=/opt/rtems \
               --with-gnu-as --with-gnu-ld --with-newlib --verbose \
               --with-system-zlib --disable-nls \
               --enable-version-specific-runtime-libs \
               --enable-threads=rtems \
               --enable-languages=c,c++
   ```

   You should again replace the ’`i386`’ with the name of the architecture for which you’re building the cross-compiler and libraries.

3. Build and install the cross-compiler and libraries by.

   ```
   make -w all install
   ```

   On Mac OS X (Darwin) you’ll have to use the command:

   ```
   make -w "CC=cc -no-cpp-precomp" all install
   ```

4. Return to the directory containing the tool and library sources.

   ```
   cd ..
   ```

2.4 Get, build and install RTEMS

2.4.1 Download the RTEMS source from the OAR web server.

At the time of writing no release of RTEMS was capable of supporting EPICS so you had to, and may still have to, work with a developer’s snapshot. The latest snapshot is available in

```
ftp://www.oarcorp.com/pub/rtems/snapshots/rtems/current/
```

The compressed *tar* archive in this directory can be downloaded using a web browser or a command-line program such as `curl` or `wget`:

```
curl --remote-name
   ’ftp://www.oarcorp.com/pub/rtems/snapshots/rtems/current/*.tar.bz2’
```

or

```
wget --passive-ftp --no-directories --retr-symlinks
   ’ftp://www.oarcorp.com/pub/rtems/snapshots/rtems/current/*.tar.bz2’
```

Depending on the type of firewall between your machine and the OAR FTP server you may need to remove the `--passive-ftp` option from the `wget` command.
When you are done you should have the compressed archive with a name something like

```
rtems-ss-20030128.tar.bz2
```

2.4.2 Unpack the RTEMS sources

Change to your RTEMS source directory and unpack the RTEMS sources by:

```
bzcat rtems-ss-20030128.tar.bz2 | tar xf -
```

This will create the directory `rtems-ss-20030128` and unpack all the RTEMS source into that directory.

2.4.3 Make changes to the RTEMS source to reflect your local conditions.

Some of the board-support-packages distributed with RTEMS may require modifications to match the hardware in use at your site. The following sections describe changes commonly made to two of these board-support-packages.

MVME167

The linker script distributed with RTEMS assumes an MVME167 with 4 Mbytes of on-board memory starting at location 0x00800000. A more common configuration is 16 Mbytes of memory starting at location 0x00000000. To reflect this configuration make the following changes to

```
rtems-ss-20030128/c/src/lib/libbsp/m68k/mvme167/startup/linkcmds

@@ -24,8 +24,8 @@
 /*
  * Declare some sizes. Heap is sized at whatever ram space is left.
  */
-_RamBase = DEFINED(_RamBase) ? _RamBase : 0x00800000;
-_RamSize = DEFINED(_RamSize) ? _RamSize : 4M;
+_RamBase = DEFINED(_RamBase) ? _RamBase : 0x0;
+_RamSize = DEFINED(_RamSize) ? _RamSize : 16M;
 _HeapSize = DEFINED(_HeapSize) ? _HeapSize : 0;
 _StackSize = DEFINED(_StackSize) ? _StackSize : 0x1000;

@@ -35,7 +35,7 @@
       This is where we put one board. The base address should be
       passed as a parameter when building multiprocessor images
       where each board resides at a different address. */
-  ram  : org = 0x00800000, l = 4M
+  ram  : org = 0x00000000, l = 16M
   rom  : org = 0xFF800000, l = 4M
   sram : org = 0xFFE00000, l = 128K
 }
```

PC-x86

A change I like to make to the RTEMS pc386 source is to increase the number of lines on the console display from 25 to 50 since I find that the output from some EPICS commands scrolls off the display when only 25 lines are present. To make this change, add the '`#define`' line shown below

```
rtems-ss-20030128/c/src/lib/libbsp/i386/pc386/start/start16.S

 +------------------------------------------------------*/

 #include <bspopts.h>
 #define RTEMS_VIDEO_80x50

 /*---------------------------------------------------------+ | Constants
```

Another change I make is to automatically fall back to using COM2: as a serial-line console (9600-8N1) if no video adapter is present. This allows the pc386 BSP to be used on conventional PCs with video adapters as well as with embedded PCs (PC-104) which have no video adapters. To make this change, add the '`#define`' line shown below

```
rtems-ss-20030128/c/src/lib/libbsp/i386/pc386/console/console.c

   */
  rtems_termios_initialize ();

#define RTEMS_RUNTIME_CONSOLE_SELECT
#ifdef RTEMS_RUNTIME_CONSOLE_SELECT
  /*
   * If no video card, fall back to serial port console
```

2.4.4 Build and install RTEMS

1. It is best to start with a clean slate. **Make very sure you're in the right directory before running the rm command!**

   ```
   cd /usr/local/source/RTEMS
   rm -rf build
   mkdir build
   cd build
   ```

2. Configure RTEMS for your target architecture:

   ```
   cd /usr/local/source/RTEMS/build
   ../rtems-ss-20030128/configure --target=i386-rtems --prefix=/opt/rtems \
       --enable-cxx --enable-rdbg --disable-tests --enable-networking \
       --enable-posix --enable-itron
   ```

   You should replace the 'i386' with the name of the architecture for which you're building RTEMS. Common alternatives are 'm68k' and 'ppc' for the Motorola M68k and Power-PC family of processors, respectively.

3. Build and install RTEMS for one or more board-support packages using the target architecture configured in the previous step. The *pc386* board support package which works on standard PC hardware would be specified as:

   ```
   gmake RTEMS_BSP=pc386
   gmake RTEMS_BSP=pc386 install
   ```

   You can build for more than one board-support package by specifying more names on the command line. For example, for the m68k-rtems architecture you can build for a generic MC68360 system and an MVME-167 system by:

   ```
   make RTEMS_BSP="gen68360 mvme167"
   make RTEMS_BSP="gen68360 mvme167" install
   ```

2.5 Get, build and install some RTEMS add-on packages (optional)

The EPICS IOC shell uses the the libtecla or GNU readline package to provide command-line editing and command history. While the IOC shell can be compiled without these capabilities I think they're important enough to warrant making the effort to download and install the extra packages.

If you don't want to bother installing the RTEMS add-on packages you must edit your EPICS local configuration file (configure/CONFIG_SITE) and add

```
EPICSCOMMANDLIN_LIBRARY = EPICS
```

before building EPICS base for RTEMS IOC targets.

2.5.1 Download the add-on package sources

The latest versions of these files are in

```
ftp://ftp.oarcorp.com/pub/rtems/snapshots/contrib/add_on_packages/
```

The compressed *tar* archive in this directory can be downloaded using a web browser or a command-line program such a `wget` (note that the wget command example has been split to make it fit on this page):

```
wget --passive-ftp --no-directories --retr-symlinks
    "ftp://ftp.oarcorp.com/pub/rtems/snapshots/contrib/add_on_packages/rtems-addon-packages-
20030128.tar.bz2"
```

Depending on the type of firewall between your machine and the OAR FTP server you may need to remove the `--passive-ftp` option from the `wget` command.
When you are done you should have the compressed archive with a name something like

```
rtems-addon-packages-20030128.tar.bz2
```

2.5.2 Unpack the add-on package sources

Change to your RTEMS source directory and unpack the RTEMS sources by:

```
cd /usr/local/source/RTEMS
bzcat rtems-addon-packages-20030128.tar.bz2 | tar xf -
```

This will unpack the source for all the RTEMS packages into a directory named

```
rtems-addon-packages-20030128
```

2.5.3 Set the RTEMS_MAKEFILE_PATH environment variable

The makefiles in the RTEMS packages use the `RTEMS_MAKEFILE_PATH` environment variable to determine the target architecture and board-support package. For example,

```
RTEMS_MAKEFILE_PATH=/opt/rtems/i386-rtems/pc386
```

will select the Intel-x86 architecture and the RTEMS pc386 board-support package.

2.5.4 Build and install the add-on packages

The *bit* script in the packages source directory builds and installs all the add-on packages. To run the script change directories to the add-on packages directory and execute:

```
sh bit
```

If you are building for more than one architecture or board-support package, you must run the *bit* script once for each variation with `RTEMS_MAKEFILE_PATH` set to the different architecture and board-support package.

2.6 Try running some RTEMS sample applications (optional)

The RTEMS build process creates some sample applications. If you're just getting started with RTEMS it's probably a good idea to verify that you can actually run a simple RTEMS application on your target hardware before trying to run a full-blown EPICS IOC application.

The actual method of loading an application into a target processor is hardware-dependent. Section 4.1.3 describes a method which may be used with RTEMS pc386 targets. To run the ’hello’ sample application, for example, follow the steps described in section 4.1.3 with the first `objcopy` command changed to;

```
i386-rtems-objcopy --output-target=binary
              /opt/rtems/i386-rtems/pc386/samples/hello.exe temp.bin
```

Chapter 3

EPICS Base

The first step in building an EPICS application is to download the EPICS 'base' source from the APS server and unpack it. The details on how to perform these operations are described on the APS web pages and will not be repeated here. Make sure you get the R3.14beta1 release or later. The alpha releases of R3.14 will not work with recent versions of RTEMS.

3.1 Specify the location of RTEMS tools and libraries

You must first let the EPICS Makefiles know where you've installed the RTEMS development tools and libraries. The default location is

```
/opt/rtems
```

If you've installed the RTEMS tools and libraries in a different location and ignored my advice to at least provide a symbolic link from

```
/opt/rtems
```

to wherever you've installed RTEMS you need to edit the EPICS configuration file

```
base-3.14.beta1/configure/os/CONFIG.Common.RTEMS
```

In this file you'll find the line

```
RTEMS_BASE=/opt/rtems
```

while will have to be changed to reflect the directory into which you installed RTEMS.

If you didn't build the RTEMS `readline` and `curses` add-on packages you should remove the `-lreadline` and `-lcurses` libraries from the `OP_SYS_LDLIBS` line.

3.2 Specify the network interface

Some RTEMS board support packages support more than one type of network interface. The pc386 BSP, for example, can be configured to use several different network interface cards. By default the EPICS network configuration for the pc386 BSP loads network drivers for all network interfaces which support run-time probing so if you've got one of these network interfaces you don't need to make any changes to the EPICS network configuration. If not, see the comments in

```
$EPICS_BASE/src/RTEMS/base/rtems_netconfig.c
```

for instructions on selecting a network interface card when building your EPICS application.

3.3 Specify the target architectures

The `base-3.14.beta1/configure/CONFIG_SITE` file specifies the target architectures and board support packages to be supported. For example, I regularly build for the following three targets:

```
CROSS_COMPILER_TARGET_ARCHS=RTEMS-gen68360 RTEMS-mvme167 RTEMS-pc386
```

If you want to build for just the RTEMS pc386 target you would change this line to

```
CROSS_COMPILER_TARGET_ARCHS=RTEMS-pc386
```

The format of the target architecture names is `RTEMS-`*bspname*, where `RTEMS-` indicates that the RTEMS development tools and libraries should be used, and *bspname* is the name of the RTEMS target architecture and board support package used back in section 2.4.4.

Some extensions still use the old configuration procedure so it's a good idea to make the same change in

```
base-3.14.beta1/config/CONFIG_SITE
```

as well.

3.4 Build EPICS base

This step is very simple. Just change directories to the EPICS `base` directory and run

```
make
```

After a while you'll end up with a working set of EPICS base libraries and tools.

Chapter 4

EPICS Applications

Now that you've built the EPICS base libraries you're ready to build and run your first EPICS application. Once you've got this application running you can forget about this tutorial and revert to using the standard EPICS documentation. You can start with your own special application or you can start with the example application that is provided with the EPICS distribution. The following sections describe the procedure to create, build and run this example application.

4.1 The EPICS example application

4.1.1 Build the example application

1. Create a new directory to hold the application source and then 'cd' to that directory.

2. Run the `makeBaseApp.pl` program to create the example application:

   ```
   makeBaseApp.pl -t example test
   makeBaseApp.pl -i -t example -a RTEMS-pc386 test
   ```

   If you get complaints about not being able to run these commands you've probably forgotten to put the 'bin' directory created in the previous section on your shell executable search path.

   The '`test`' in the two `makeBaseApp.pl` commands can be replaced with whatever name you want to give your example application. The '`RTEMS-pc386`' can be replaced with whatever target architecture you plan to use to run the example application.

3. Build the example application by running

   ```
   make
   ```

4.1.2 Install the EPICS IOC files on the TFTP server

The IOC application build process produces a set of IOC shell commands in the `st.cmd` file in the `iocBoot/ioctest` directory. If you chose an application name different than `test` in the previous step, the directory name will change accordingly.

RTEMS uses the trivial file transfer protocol (TFTP) to read these IOC shell commands and database information. The RTEMS startup code first changes directories to `/epics/`*hostname*`/` within the TFTP server, where *hostname* is the Internet host name of the IOC. The startup code then reads IOC shell commands from the `st.cmd` script in that directory.

The name (st.cmd) and location of the startup script are fixed from the IOCs point of view so it must be installed in the corresponding location on the TFTP server. Many sites run the TFTP server with an option which changes its root directory. On this type of system you'll have to copy the startup script to the /epics/*hostname*/ directory within the TFTP server's root directory. On my system, for example, the TFTP server runs with its root directory set to /tftpboot so I install the startup script for the IOC who's name is norumx1 by

```
cp iocBoot/iocexample/st.cmd /tftpboot/epics/norumx1/st.cmd
```

On systems which do not change the TFTP server's root directory you would leave off the leading /tftpboot component of the path name.

The application build process also creates db and dbd directories in the top-level application directory. These directories and their contents must be copied to the IOC's directory on the TFTP server. On my system the command to install the files for the norumx1 IOC is

```
cp -r db dbd /tftpboot/epics/norumx1
```

Appendix B contains a shell script which automates the copying and editing operations described in this section.

4.1.3 Run the example application on an RTEMS IOC

Everything's now ready to go. The only item left is arranging some way of loading the RTEMS/EPICS application executable image into the IOC. There are many ways of doing this (floppy disks, PROM images, etc.), but I find using a BOOTP/DHCP/TFTP server to be the most convenient. The remainder of this section describes how I load executables into my RTEMS-pc386 IOCs. If you're running a different type of IOC you'll have to figure out the required steps on your own. The RTEMS documentation may provide the required information since an EPICS IOC application is an RTEMS application like any other. EPICS RTEMS IOC applications use BOOTP to obtain their network configuration at run-time, so you'll need a BOOTP/DHCP server on your network in any case. As well, EPICS RTEMS IOC applications use NTP to set the IOC clock so an NTP server is also required. If the RTEMS IOC doesn't receive a time-synchronization packet from an NTP server the IOC time will be set to January 1, 2001.

I have installed an EtherBoot bootstrap PROM image obtained from the 'ROM-o-matic' server (ttp://www.rom-o-matic.net/) on the IOC network interface cards. Unfortunately the network boot process can not directly load the executable image produced by the EPICS/RTEMS IOC application build process. The executable image must be merged with some 16-bit startup code to produce a network boot image which is then downloaded and executed by the RTEMS-pc386 IOC.

1. Convert the application executable image to simple binary format:

   ```
   i386-rtems-objcopy --output-target=binary bin/RTEMS-pc386/example temp.bin
   ```

2. Merge the application binary and the 16-bit startup code to form the network boot image:

   ```
   /opt/rtems/i386-rtems/pc386/build-tools/bin2boot example.bt 0x00097E00 \
           /opt/rtems/i386-rtems/pc386/lib/start16.bin 0x00097C00 0 \
           temp.bin 0x00100000 0
   ```

3. Copy the network boot image to the BOOTP/TFTP directory. The actual location where you place the executable image depends on the configuration of your BOOTP/DHCP server. I put the executables in the IOC's directory on the TFTP server in a bin subdirectory as follows:

   ```
   cp example.bt /tftpboot/epics/norumx1/bin/example.bt
   ```

Appendix A

Script to get and build the cross-development tools

If you're feeling brave you can turn loose the following script. It attempts to download, unpack, configure, build and install the GNU cross-development tools and libraries for one or more target architectures. To use the script, edit the `ARCH="i386"` line to reflect the architectures you wish to support, then

```
sh getAndBuildTools.sh
```

```
#!/bin/sh

#
# Get, build and install the latest cross-development tools and libraries
#

#
# Specify the architectures for which the tools are to be built
# To build for multiple targets: ARCHS="m68k ppc i386"
#
ARCHS="i386"

#
# Where to install
#
PREFIX=/opt/rtems

#
# Specify the versions
#
GCC=gcc-3.2.1
BINUTILS=binutils-2.13.2.1
NEWLIB=newlib-1.11.0
GCCDIFF=20021209
NEWLIBDIFF=20030203

#
# Where to get the GNU tools
```

```
#
RTEMS_SOURCES_URL=ftp://www.oarcorp.com/pub/rtems/snapshots/c_tools/source
RTEMS_BINUTILS_URL=${RTEMS_SOURCES_URL}/${BINUTILS}.tar.bz2
RTEMS_GCC_URL=${RTEMS_SOURCES_URL}/${GCC}.tar.gz
RTEMS_NEWLIB_URL=${RTEMS_SOURCES_URL}/${NEWLIB}.tar.gz
RTEMS_GCC_DIFF_URL=${RTEMS_SOURCES_URL}/${GCC}-rtems-${GCCDIFF}.diff
RTEMS_NEWLIB_DIFF_URL=${RTEMS_SOURCES_URL}/${NEWLIB}-rtems-${NEWLIBDIFF}.diff

#
# Allow environment to override name of make program
#
MAKE="${MAKE:-make}"

#
# Get the source
# If you don’t have curl on your machine, try using
#     wget --passive-ftp --no-directories --retr-symlinks <<url>>
# If that doesn’t work, try without the --passive-ftp option.
#
getSource() {
    curl --remote-name "${RTEMS_BINUTILS_URL}"
    curl --remote-name "${RTEMS_GCC_URL}"
    curl --remote-name "${RTEMS_GCC_DIFF_URL}"
    curl --remote-name "${RTEMS_NEWLIB_URL}"
    curl --remote-name "${RTEMS_NEWLIB_DIFF_URL}"
}

#
# Unpack the source
#
unpackSource() {
    rm -rf "{$BINUTILS}"
    bzcat "${BINUTILS}.tar.bz2" | tar xf -
    for d in "${BINUTILS}"*.diff
    do
        if [ -r "$d" ]
        then
            cat "$d" | (cd "${BINUTILS}" ; patch -p1)
        fi
    done

    rm -rf "${GCC}"
    zcat "${GCC}.tar.gz" | tar xf -
    for d in "${GCC}"*.diff
    do
        if [ -r "$d" ]
        then
            cat "$d" | (cd "${GCC}" ; patch -p1)
        fi
    done
    rm -rf "${NEWLIB}"
    zcat "${NEWLIB}.tar.gz" | tar xf -
    for d in "${NEWLIB}"*.diff
    do
        if [ -r "$d" ]
        then
            cat "$d" | (cd "${NEWLIB}" ; patch -p1)
        fi
    done
    (cd "${GCC}" ; ln -s "../${NEWLIB}/newlib" newlib)
}

#
# Build
#
build() {
    PATH="${PREFIX}/bin:$PATH"
    for arch in $ARCHS
    do
        rm -rf build
        mkdir build
        cd build
        "../${BINUTILS}/configure" "--target=${arch}-rtems" "--prefix=${PREFIX}"
        ${MAKE} -w all install
        cd ..

        rm -rf build
        mkdir build
        cd build
       "../${GCC}/configure" "--target=${arch}-rtems" "--prefix=${PREFIX}" \
            --with-gnu-as --with-gnu-ld --with-newlib --verbose \
            --with-system-zlib --disable-nls \
            --enable-version-specific-runtime-libs \
            --enable-threads=rtems \
            --enable-languages=c,c++

        case "‘uname‘" in
            Darwin) ${MAKE} -w "CC=cc -no-cpp-precomp" all ;;
            *)      ${MAKE} -w all ;;
        esac
        ${MAKE} -w install
        cd ..
    done
}

#
# Do everything
#
# Comment out any activities you wish to omit
#
set -ex
```

```
getSource
unpackSource
build
```

Appendix B

Script to install EPICS IOC application files

Here’s the script I use to copy the required files from my application build directory to the TFTP server directory. The script should be run from the top-level application build directory with the names of the target IOCs as arguments. For example,

```
sh InstallApp norumx1
```

installs the startup script and database files for an IOC whose name is `norumx1`.

```
#!/bin/sh

#
# Install EPICS/RTEMS application
#

TFTPDIR=/tftpboot/epics

#
# Copy files to TFTP region
#
for target in "$@"
do
dir="$TFTPDIR/$target"
mkdir -p "$dir" "$dir/dbd" "$dir/db"
cp -r dbd db "$dir"
cp iocBoot/*/st.cmd "$dir/st.cmd"
done

echo ’Remember to install bootable image, too’
```

You will have to modify this script to match the location of your TFTP server’s root directory.

The script does not attempt to install the application executable image since different target architectures require different methods to install the executable image.